Profiling Energy Usage of Web-Service Applications on Clusters

Yangyang Liu, Member, IEEE[*], Mohammed Alghamdi, Member, IEEE[†], Wei-Shinn Ku, Member, IEEE[‡], Yi Zhou, Member, IEEE[§], Shubbhi Taneja, Member, IEEE[¶], Xiao Qin, Member, IEEE[‖]

Abstract—Energy saving is rapidly becoming one of the hottest topics in technology field within recent decades. With the development of technology, it brings a sheer increasing trend of data and the growth scale of clusters and data centers. Meanwhile, it also raises another essential issue into the path: energy cost. In this paper, we are diving into this key issue and evaluating energy-efficiency based on TPC-W benchmark: a notable web transaction e-commerce benchmark. We simulate the web transaction with different database sizes and collect the energy data by KILL-A-WATT. Also, we deploy this setup on four different cluster systems: PC nodes and wimpy nodes, and two different heterogeneous systems: using PC as front server and wimpy as Database server, and using wimpy as Web server and PC as Database server. Energy result demonstrates different characteristics among them, which can give lightening advice for future works in data center.

1 Introduction

Although various energy conservation techniques have been proposed to reduce energy cost of clusters, little attention has been paid to energy profiling of web-service applications on heterogeneous clusters. In this study, we analyze the energy consumption of web services running on heterogeneous clusters, where high-performance nodes and wimpy nodes are integrated.

The following three factors motivate us to investigate energy usage of web-service applications running on heterogeneous clusters.

- High energy cost of clusters housed in modern data centers.
- Popularity of web services supporting e-commerce applications.
- Feasibility of applying wimpy nodes to conserve energy in clusters.

Motivation 1. Nowadays, there is a growing demand of computing in data centerss. Recently, a report revealed that the total electricity usage of data centers from 2005 to 2010 dramatically grew compared with the growth from 2000 to 2005 [1]. Growing evidence shows that energy efficiency of clusters must be improved to reduce the operation cost of data centers.

Motivation 2. An increasing number of Web-service applications are running on clusters housed in data centers. Sample web-service applications include YouTube, Facebook, Twitter, and to name just a few. How to develop energy-efficient web-service applications running on cluster is still an open issue. We start to address this issue by profiling energy usage of web services on clusters.

Motivation 3. In the past decade, a handful of energy-saving techniques have been developed to improve energy efficiency of large-scale clusters. For example, FAWN (Fast Array of wimpy Nodes) is a new cluster architecture with low-power embedded CPUs [2]. We are motivated by advanced wimpy nodes to investigate the possibility of substituting conventional high-performance nodes with wimpy nodes to build energy-efficient clusters running web-service applications.

In this study, we conduct energy profiling research on a real-world e-commerce transaction system, where energy profiles inspire developers to design and implement novel energy conservation techniques. We investigate energy efficiency of various hardware configurations, thereby reducing energy consumption by configuring system setups. In our experiments, we examine a total of four scenarios.

The rest of the paper is organized as following. Section 2 summarizes the prior studies related to energy-efficient computing.. Section 3 introduces the background and basic information on setup configurations. The profiling results of homogeneous and heterogeneous configurations can be found in Sections 4. Section 5 makes conclusions and offers future work directions.

2 Related Work

In the past decade, a wide range of energy conservation techniques were proposed to reduce operation cost of large-scale clusters in data centers. Existing schemes make an effort to make good tradeoffs between energy efficiency and performance. Existing power management strategies fall into two camps, namely, static power management (SPM) and dynamic power management systems (DPM) [3]. For SPM camp, the main idea of this approach is to utilise low-power components, such as flash memory and low energy consumption in Gordon clusters [4]. Different from the above studies that were focused on optimizing energy efficiency in clusters, our research pays attention to energy usage profiling of web applications running on clusters.

A handful of systems adopt low-power components to improve system energy efficiency. For example, the FAWN architecture integrates low-power embedded CPUs with local flash storage, thereby balancing computation and I/O capabilities to enable efficient and massively parallel access to data [2]. Our work is distinctive to the above studies in the way that we investigate the energy efficiency of wimpy clusters under e-commerce workload conditions. We discover that wimpy clusters are promising platforms to energy-efficiently run web applications.

3 System Framework And Experiment Setup

3.1 Frameworks

Bearing the design motivations in mind, we develop a testbed to study energy efficiency of web services running on clusters. The testbed embraces the multiple-tier architecture design to resemble real-world flexible and scalable systems.

The multiple tier architecture in general and the three-tier architecture in particular are wildly adopted by clusters housed in modern data centers. Fig 1 depicts the layout of the three-tier architecture and the detailed software framework of our testbed, where clients in tier one submit requests to front or Web servers in the middle tier, and the requests retrieve data managed by data servers in tier three.

To investigate energy profiles of web services, we deploy the TPC-W benchmark on the three-tier testbed to dynamically emulate real-time e-commerce transactions. The testbed processes the web workload of a book retail website, where multiple web browsers are running to emulate multiple clients [5]. The testbed utilizes emulated browsers (EB) to simulate multiple active users shopping online. The primary matrics of TPC-W are the Web Interactions Per Second or *WIPS*, which has two variants, namely *WIPSb* and *WIPSo*. *WIPSb* represents the *WIPS* performance of the browsing process; whereas *WIPSo* refers to the *WIPS* performance of the ordering procedure.

We apply these two metrics to study the performance of 14 types of transaction requests in the benchmark. These requests types are grouped into three mix camps, namely, the browsing mix, shopping mix, and ordering mix. We substantially increase the database size to resemble modern big-data applications running on clusters by increasing ITEM table size.

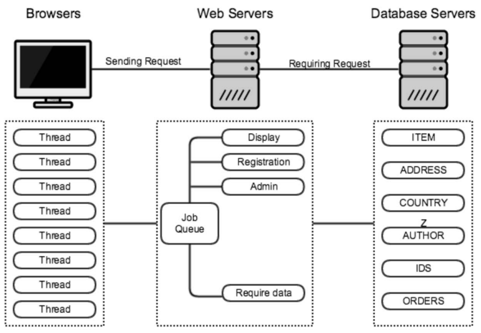


Fig. 1. The software framework about the TPC-W benchmark.

3.2 Setup and Configurations

We delineate the experiment setup, where four configurations are evaluated. We deploy the PC and wimpy nodes as the web server and database server in our three-tier architecture, respectively. Thus, the four setup configurations include:

- **Homogeneous Cluster 1:** PC Web Server and PC Database Server.
- **Homogeneous Cluster 2:** Wimpy Web Server and Wimpy Database Server.
- **Heterogeneous Cluster 1:** PC Web Server and Wimpy Database Server.
- **Heterogeneous Cluster 2:** Wimpy Web Server and PC Database Server.

We refer to the first two configurations as homogeneous setups (see the results in Sections 4.1); whereas the third and fourth configurations are considered as heterogeneous setups (see the results in Section 4.2).

3.3 Experimental Methodology

In all the four aforementioned configurations, clients keep randomly sending requests governed by the scripts. Approximately 95% requests in the browsing mix are browsing requests, which involve searching and displaying contents. We concurrently run a hundred threads to simulate distinct clients or customers, which can help to resemble real-world web service scenarios. We also follow the specification of the TPC-W benchmark [6] by determining the sleep time interval between two consecutive client requests. While we are simulating requests handled by the web-service cluster in the three-tier architecture, the electricity monitors are incorporated to measure energy cost. This accumulative energy measures are collected as a total energy cost E with the unit of kWh. Let N denote the total number of requests issued to the web-service cluster in each experiment. We measure the energy efficiency in terms of energy consumption per request.Thus, we apply (1) to quantify energy cost of each configuration under various workload conditions. In the subsequent sections, we use the term *energy cost* and EP interchangeably.

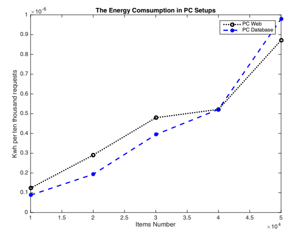
$$EP = \frac{E}{N}. \tag{1}$$

4 Performance Evaluation

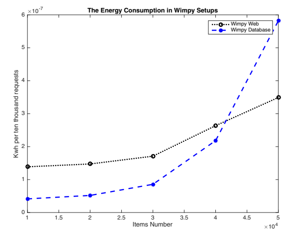
Now we are in a position to present experimental results of the tested web-service cluster configured by four setups (see also Section 3.2). In Sections 4.1, we evaluate the energy efficiency and performance of the homogeneous PC and wimpy clusters. Finally, we present the experimental results of the heterogeneous clusters in Section 4.2.

4.1 Two Homogeneous Clusters

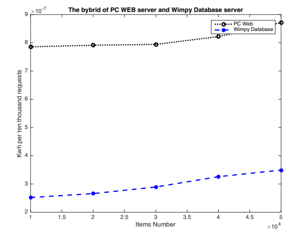
Our primary aim is collecting from experiments and graphing the dynamic figures of the whole setup. We are also trying to change the database size by modifying ITEM number, which is a frequently used table in this mix. We keep the same ITEM contents in the ITEM table but simply populate more items into ITEM table for a different traversal time. In terms of ITEM numbers, we test energy with $1{\times}10^4$, $2{\times}10^4$, $3{\times}10^4$, $4{\times}10^4$ and $5{\times}10^4$ numbers. The energy trends are demonstrated in Fig. 2 (a) and (b). Another thing needs to be monitored is the performance for job handling, so we



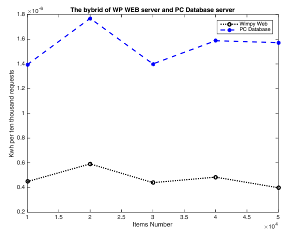
(a) PC homogeneous cluster



(b) Wimpy homogeneous cluster

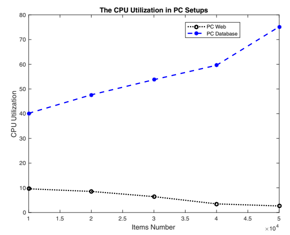


(c) PC Web and Wimpy Database Servers

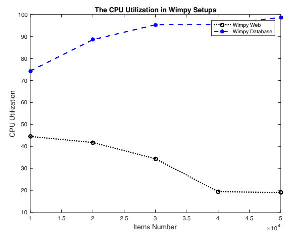


(d) Wimpy Web and PC Database Servers

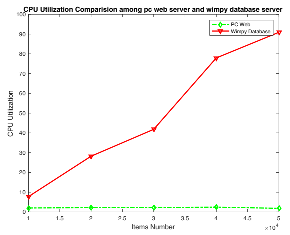
Fig. 2. The energy trends of the web and database servers of all 4 cluters.



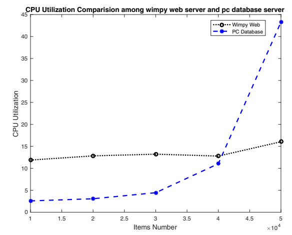
(a) PC homogeneous cluster



(b) Wimpy homogeneous cluster



(c) PC Web and Wimpy Database Servers



(d) Wimpy Web and PC Database Servers

Fig. 3. Impact of the table size on CPU utilization among all four clusters.

keep monitoring and recording the CPU utilization every 60 seconds. Fig. 3 (a) and (b) is telling the trends about the CPU utilization on front and back servers, this will be a clear overview for burdens on two nodes.

In all two homogeneous clusters, Fig. 2 (a) and (b) shows that populating more item numbers gives rise of increased energy cost in both web and database servers. When the number of items is below $4{\times}10^4$, the Web server's energy consumption is higher than that of the database server. This trend is reasonable, because the JBoss middleware imposes extra energy cost to the Web server. Note that JBoss - playing an application server role - facilitates the communication between clients and the database. The front Web server receives requests; the Web server also processes any job that does not access data from the backend server (e.g., such requests include home page accesses and ordering display). Processing these types of requests lead to an increased energy consumption in the Web server even when the number of items is relatively small (i.e., fewer than $4{\times}10^4$).

When the number of items is larger than $4{\times}10^4$, further increasing $ITEM$ dramatically pushes up the energy consumption of the database server. As result, the energy consumption of the database server exceeds that of the Web server. The significantly increased energy consumption at the database server is attributed to the increased CPU utilization. For example, Fig. 3 (a) and (b) reveals that the database server's CPU utilization is consistently growing when we increase the number of items; in contrast, the CPU utilization of the Web server is slightly dragged down with an increasing $ITEM$.

Fig. 3 (a) and (b) also shows that regardless of the $ITEM$ value, the CPU utilization of the database server is always higher than that of the Web server. When the $ITEM$ becomes large, the CPU utilization gap between the database and Web servers widens. In what follows, we explain the reasoning behind such a widened CPU-utilization gap.

Our analysis suggests that the database server's high CPU utilization is contributed by the nature of the browsing mix requests, which demand excessive CPU computing resources in the back-end nodes. We observe that 95% of the total requests belong to one of the browsing mix requests, among which the product-detail, search-request, and search-results requests are proactively accessing data residing in the back-end database. Adding more items in the database enlarges the item table size, which inevitably extends item-table traversing time. The long traversal times in turn lead to slow request response times of the Web server, thereby keeping an increased number of requests in a queue in the database server. Consequently, the front-end Web server maintains low CPU utilization thanks to (1) the database's slow response and (2) a long waiting queue at the back-end server.

4.2 Heterogeneous Clusters

4.2.1 Overall Performance Comparisons

In this subsection, we aim to propose an energy-efficient way of building heterogeneous clusters by investigating the energy efficiency of two types of heterogeneous web-service clusters. In the first configuration is comprised of PC web and wimpy database servers; whereas the second heterogeneous system contains wimpy web and PC database servers. For each heterogeneous cluster, we measure energy consumption and CPU utilization under web-service workload conditions. Due to the unbalanced energy and performance efficiency between PC and wimpy servers, the tested heterogeneous clusters exhibit distinctive energy and performance behaviors compared with homogeneous counterparts.

4.2.2 PC Web and Wimpy Database Servers

In the scenario of the first heterogeneous setup (i.e., PC web and wimpy database servers), the job queue in the PC Web server is almost empty during the course of the entire testing experiment. The evidence can be found in Fig. 3 (c), which reveals that the CPU utilization of the PC web server stays at a very low level. Thus, the load imposed on the PC web server is light, making the PC web server sit idle for the majority of the time.

In contrast, Fig. 3 (c) illustrate that CPU utilization of the wimpy database server rises dramatically, indicating that the wimpy server becomes heavily loaded when the number of items is large. Such CPU-utilization trend pictures a fact that the performance bottleneck lies in the back-end wimpy nodes. The performance gap between these two types of servers largely depends on the computing capacity of the internal processors.

Upon the arrivals of requests, the front-end PC server easily handles all the displaying requests. All the other requests requiring detailed item information are placed by the PC web server into a waiting queue, which is connected to the back-end wimpy database server. The wimpy server is unable to keep the pace with the fast speed of the PC server; this problem is worsened when the wimpy server's processing time of each request (e.g., data traversing time) is large. Under heavy workload conditions, the communication between the PC web and wimpy database servers is blocked, meaning that a significant number of requests are waiting at the front-end server due to the saturated wimpy server.

An important conclusion drawn from Fig. 2 (c) is that the first heterogeneous system is energy-efficient under light load (e.g., ITEM is smaller than $2{\times}10^4$). The energy efficiency of the PC web and wimpy database server noticeably drops under heavy workload (e.g., ITEM is larger than $3{\times}10^4$). The energy consumption per request of the two types of servers is almost flat before the wimpy database server becomes a performance bottleneck. Once the wimpy server is overly loaded, the energy efficiency of both web and database systems is downgrading.

4.2.3 Wimpy Web and PC Database Servers

Now we offer an analysis on the energy efficiency of the second heterogeneous system powered by wimpy web and PC database servers. Fig. 2 (d) reveals that the second heterogeneous system exhibits an unpredictable energy efficiency when it comes to web-service applications. The high-performance PC database server is capable of handling a large number of requests issued by the front-end wimpy server, implying that the PC database server never becomes the system performance bottleneck at the back end.

Fig. 3 (d) show that the CPU utilization of the PC database server marginally grows from 2% to 5% when the number of items varies from $1{\times}10^4$ to $3{\times}10^4$. Then, the database server's CPU utilization jumps to 40% when the number of items is as large as $5{\times}10^4$. One attributing factor is that the local cache in the wimpy web server is relatively small, which is insufficient to cache a massive amount of data for future requests (e.g., browsing and searching). Previously cached data are likely to be repeatedly evicted from the wimpy server in order to accommodate recent requests.

The second heterogeneous system's CPU trend of the wimpy web server is very different from that of the first one. Fig. 3 (c) evidently indicates that the wimpy web server's CPU utilization slightly increases when the number of items increases, whereas the CPU utilization decreases in the other three configurations. It is arguably true that the overall performance of the second heterogeneous cluster largely depends on the front-end wimpy node. Requests accessing the database are placed in a job queue residing in the PC server, waiting for responses from the database hosted on the PC server. The wimpy front-end node has to independently handle displaying and ordering requests without interacting with the database server. When the wimpy web server exhibits a long request queue, the wimpy node is unable to handle incoming requests in a timely manner.

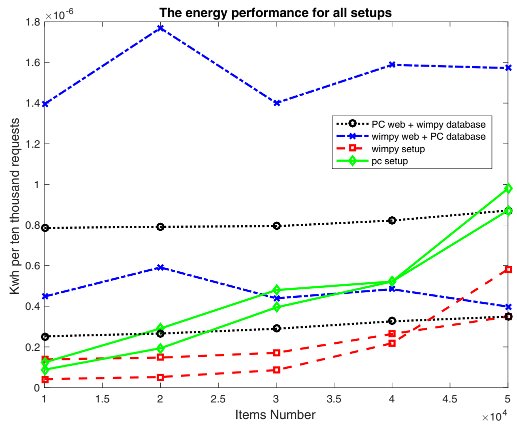


Fig. 4. An overall comparison for all setups.

4.3 Put It All Together

Fig. 4 illustrates the energy cost trends of all the four tested configurations. We draw the following three observations by comparing the four configurations.

First, the cluster coupled with the wimpy web and PC database servers delivers the worst performance among all the solutions. If improving energy efficiency is the sole purpose, the homogeneous wimpy cluster is undoubtedly a winner under all workload conditions. In the case of a homogeneous cluster, the processing capacity of front-end servers is on par with that of back-end servers.

Second, although the two heterogeneous systems have identical hardware components, they exhibit distinct performance and energy efficiency. The system equipped with the wimpy web and PC database servers becomes the worst choice, whereas the PC-web-wimpy-database system is considered a competitive candidate.

Last, when it comes to overall system performance, the PC-web-wimpy-database system is a good choice. The only downside for such a configuration is that the energy efficiency is low when data size is small due to unbalanced load between the PC web and wimpy database servers. Fortunately, when the data size is large (e.g., the number of items increases to $4{\times}10^4$ and $5{\times}10^4$), this heterogeneous system can not only speedup the request process compared with the two homogeneous clusters, but also be more energy efficient than the homogeneous PC cluster.

5 Conclusions and Future Work

Evidence shows that for complex data processing workloads, a large-scale wimpy cluster may not be as energy-efficient as a small-scale traditional cluster [7]. We believe that job scheduling policies, heterogeneity configurations, workload profiling, and scaleup analysis are promising ways to optimize performance of heterogeneous clusters powered by energy-efficient wimpy nodes.

Among all the four tested configurations, the cluster containing PC web and wimpy database servers makes the best tradeoff between performance and energy efficiency. To speedup back-end processing performance, one may increase the number of wimpy nodes of the wimpy cluster serving as a database system. Large-scale wimpy clusters have a high reliability compared with their low-scale counterparts. With a large-scale wimpy database cluster in place, we plan to design a node partitioning strategy to dynamically partition the wimpy cluster into a set of small-scale sub-clusters. We also will develop a scheduler to dispatch requests to the multiple sub-clusters in a way to maximize energy savings while maintaining good performance.

Recall that Fig. 4 shows that the heterogeneous cluster with PC web and wimpy database servers consumes more energy than the two homogeneous systems. In the future, we will address this drawback by investigating a heterogeneous computing setup where the back-end database servers are powered by both energy-efficient wimpy and high-performance PC servers. In our design, we plan to have the wimpy nodes store data items that optimize the energy efficiency of the wimpy database servers; PC database nodes manage data items that make PC servers offer energy savings.

Acknowledgments

The authors would like to thank Mojen Lau for proofreading this manuscript. This research was supported by the U.S. National Science Foundation under Grants CCF-0845257 (CAREER), CNS-0917137 (CSR), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS).

References

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, 2011.

[2] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, "FAWN: a fast array of wimpy nodes," in *Proceedings of the 22nd ACM Symposium on Operating Systems Principles 2009, SOSP 2009, Big Sky, Montana, USA, October 11-14, 2009*, 2009, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/1629575.1629577

[3] G. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Computing*, vol. 16, no. 1, pp. 3–15, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10586-011-0171-x

[4] A. M. Caulfield, L. M. Grupp, and S. Swanson, "Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2009, Washington, DC, USA, March 7-11, 2009*, 2009, pp. 217–228. [Online]. Available: http://doi.acm.org/10.1145/1508244.1508270

[5] "Tpc-w benckmark specification," http://www.tpc.org/tpcw/spec/tpcw_v1.8.pdf.

[6] D. A. Menascé, "TPC-W: A benchmark for e-commerce," *IEEE Internet Computing*, vol. 6, no. 3, pp. 83–87, 2002. [Online]. Available: http://dx.doi.org/10.1109/MIC.2002.1003136

[7] W. Lang, J. M. Patel, and S. Shankar, "Wimpy node clusters: what about non-wimpy workloads?" in *Proceedings of the Sixth International Workshop on Data Management on New Hardware, DaMoN 2010, Indianapolis, IN, USA, June 7, 2010*, 2010, pp. 47–55. [Online]. Available: http://doi.acm.org/10.1145/1869389.1869396